APPENDIX B

```
%{
/*************************************************************************
**
 *
 *   Copyright 1996, 1997, 1998 - A-LIFE Medical, Inc. - All Rights
Reserved.
 *
 *   Program: LifeCode Medical Coding System
 *
 *   File: c_tagger.lex
 *
 *   Description:
 *     This file defines the CPT tagger of the diagnosis sections.
 *
 *

*************************************************************************
**/

#include <iostream.h>
#include "keys.h"

#ifdef LEXER
#include <strstrea.h>
#include <rw/cstring.h>
class CPTCoderClass {

  private:

   RWCString text_str;
   RWCString key_str;

  public:

   void add_parse_text(char *orig, char *term) {
      text_str += term;
      text_str += " ";
   }

   void add_parse_key(int id, char *key) {
      text_str += "| ";
      ostrstream ostr;
      char *id_str;
      ostr << id << ends;
      id_str = ostr.str();
      key_str += id_str;
      key_str += key;
      key_str += " ";
      delete id_str;
   }

   void parse_procedure(void) {
      cout << text_str << endl;
      cout << key_str << endl;
```

```
      text_str = "";
      key_str = "";
   }

   void set_admit(void) { }
   void set_pa(void) { }
   void set_xray_confirm(void) { }
   void set_xray_disconfirm(void) { }
   void add_break(void) { }
   void add_parse_negation(void) { text_str += "NEG "; }
   void add_parse_instruct(void) { text_str += "INS "; }
   void add_parse_loc(int loc) { text_str += "LOC "; }
   void add_parse_ordinal(int ord) { text_str += "ORD "; }
   void add_parse_severity(int sev) { text_str += "SEV "; }
   void add_parse_cardinal(double card) { text_str += "CARD "; }
   void add_parse_size(double size) {  text_str += "SIZE"; }
   void add_parse_size_scale(int size) {  text_str += "SIZE_SCALE"; }

   void add_parse_temp_cardinal(double card) { text_str += "TEMP_CARD
"; }
   void add_parse_temp_rel(int rel) { text_str += "TEMP_REL "; }
   void add_parse_temp_scale(int scale) { text_str += "TEMP_SCALE "; }

   void add_parse_bodysurf_cardinal(double card) { text_str +=
"BODY_CARD "; }
   void add_parse_bodysurf_rel(int rel) { text_str += "BODY_REL "; }

   void add_parse_probability(int prob) { text_str += "PROB "; }
   void add_parse_evidence(int evid) { text_str += "EVID "; }
   void add_parse_diag_ordinality(int ord) { text_str += "DIAG_ORD "; }
   void add_parse_status(int stat) { text_str += "STAT "; }
   void add_parse_etiology(int eti) { text_str += "ETI "; }
   void add_parse_diagnostic(int diag) { text_str += "DIAG "; }

   CPTCoderClass(): text_str(""), key_str("") {
   }

   ~CPTCoderClass() {
      cout << text_str << endl;
      cout << key_str << endl;
   }
};
#endif

#ifndef LEXER
#include "ccoder.h"
#include "cparse_item.h"
#undef ECHO
#define ECHO 1
#undef YY_INPUT
#define YY_INPUT(buf,result,max_size) \
    CPTCoder.code_lexer.get(buf, max_size, EOF); \
    result = CPTCoder.code_lexer.gcount(); \
    CPTCoder.code_lexer.accumulate(result);
#endif

extern CPTCoderClass CPTCoder;
```

```
extern int yylex();
int yy_temporal = 0;
int yy_bodysurf = 0;
int yy_measure = 0;
%}

SP          ([ ]+)
DISCARD           (the|a|his|her|[\"]){SP}

DISCHARGE   (((not{SP}(be{SP}|being{SP})?)|no{SP}|\\
(request(ing|ed|s)?{SP}))(admit|admitted|admitting|admission))
SENT        (was{SP}|will({SP}be)?{SP})?\\
(sent|send|take|taken|took|move|moved|moving)({SP}(by|to|for|into{SP}))
?
ADMIT
      (was{SP}|will({SP}be)?{SP})?(admit|admitted|admitting|admission)(
{SP}\\
(by|to|for|into{SP}))?
SURGERY           ({SENT}|{ADMIT})surgery
FIELD_WORK  ((in|at){SP}(the{SP})?\\
(field|ambulance|life{SP}flight|helicopter|police{SP}car|squad{SP}car|n
ursing{SP}home)|\\
(by{SP})?(the{SP})?(they|them|emt|ams|medics?|paramedics?|rescue{SP}squ
ad|ambulance)|\\
before{SP}arriv(al|ing))
DOCTOR            (neurologist|orthopod|ortho(pod|pedic)?{SP}\\
(surge(ry|on)|team|tech(nicians?)?|technicians?))
PHY_ASST    (PA|physician'?s?{SP}assistant|p{SP}\.{SP}a{SP}\.)
RAD_READ1
      (radiolog(ist|y){SP}(read|interpret(ed)?)|(per|read|interpret(ed)
?){SP}\\
(as{SP}negative{SP})?((by|of){SP})?(the{SP})?radiolog(ist|y))
RAD_READ2
      ((per|read|interpret(ed)?){SP}as{SP}negative{SP}for{SP}((fracture
{SP}\\
(and|or){SP}dislocate{SP})|(fracture{SP})|(dislocate{SP}))((by|of){SP})
?(the{SP})?\\
radiolog(ist|y))
ED_READ1
      (radiolog(ist|y){SP}((was|is){SP})?(off{SP}duty|not{SP}available|
\\
unavailable|not{SP}on{SP}duty)|(not?[
]radiolo)|radiolog(y|ist){SP}(was{SP})?(not|un))
ED_READ2
      ((E\.?{SP}?[DR]\.?{SP}physician|emergency{SP}department{SP}physic
ian|I)\\
{SP}(read|interpret(ed)?)|(per|read|interpret(ed)?){SP}((by|of){SP})?(t
he{SP})?\\
(E\.?{SP}?[DR]\.?{SP}physician|emergency{SP}department{SP}physician|me|
myself))
SELF
      (I{SP}|I{SP},|me{SP}|by{SP}me|myself|by{SP}myself|I{SP}myself)
PLAN        (plan:|plan{SP}|plan{SP}to|plan{SP}is{SP}to|\\
may{SP}be{SP}necessary|may{SP}be{SP}needed|may{SP}(ultimately|eventuall
y|\\
some[ ]?day|some[
]?time){SP}be{SP}necessary|may{SP}(ultimately|eventually|\\
```

```
some[ ]?day|some[ ]?time){SP}be{SP}needed)
CONSIDER
      (considered{SP}|consider{SP}|consideration{SP}|deemed{SP}|\\
deem{SP}|discuss{SP}|discussed{SP}|discussion{SP})
CANCEL
      (cancelled{SP}|cancel{SP}|cancell?ing{SP}|discontinued?{SP})
REFUSE            (refused{SP}|refusing{SP}|refuses{SP}|refuse{SP})
WILL        (will{SP})


PATIENT                  (she{SP}|he{SP})

POSSIBLE_11       ("unable to altogether rule out"|\\
"unable to completely rule out"|"can not rule out"|"cannot rule out"|\\
"cannot altogether exclude"|"cannot completely exclude"|"can not
exclude"|\\
"could also be"|"could be"|"evidence
of"|"possible"|"possibly"|"questionably"|\\
"questionable"|"question"|"suspect"|"doubt")
POSSIBLE_10       ("suspected")
POSSIBLE_01       ("doubtful"|"doubted")
PROBABLE_11       ("probable"|"probably"|"likely"|"clearly")
PROBABLE_10       ("apparent")

RULE_OUT_11       ("rule out"|"no evidence of")
RULE_OUT_10       ("unfounded")

RISK_OF_11        ("low risk for"|"low risk of"|"at-risk for"|"at risk
for"|"at-risk of"|"at risk of"|"risk of")

VS_0011                  ("vs. "|"vs "|"versus"|"verses")

NOT_IN_ER_01             (no(t|ne)?[ ]eviden(t|ce(d)?)[ ]in[
](emergency|ER)\\
[ ](room|department|dept(\.)?)

EVIDENCED_11             ("evidenced")
EVIDENCE_01       ("evidence ")
ALLEGED_00        ("alleged by patient"|"per patient"|"by patient")
ALLEGED_11        ("patient alleged"|"alleged"|"patient
alleges"|"alleges"|\\
"allege "|"patient also reports"|"patient also reported"|"patient
reports"|\\
"patient reported"|"patient also states"|"patient also stated"|"patient
states"|\\
"patient stated"|"stated "|"reported ")
ATTEMPT                  (attempt(ed[ ]to[ ])?)
DENIED_00         ("denied by patient")
DENIED_11         ("patient denied"|"denied"|"patient
denies"|"denies"|"deny")

STATUS_POST_11           ("status post")
POST_11                  ("post "|"following"|"after")
HISTORY_OF_11            ("history of"|"historic"|"history")
HISTORY_OF_00            ("by history"|"in the past")
OLD_11                   ("old "|"previous ")
```

```
PRIMARY_11          ("primary")
SECONDARY_00              ("secondary to")
NUMBER              ([0-9]+(\.[0-9]+)?)
NUMBER_11           ((number[ ]|[ ]no.[ ]|#[ ]?){NUMBER})
SECONDARY_TO              ({SECONDARY_00}[ ]{NUMBER_11})
SECONDARY_11              ("secondary")

NONDISPLACED_10     ("nondisplaced")
SUBACUTE_11         ("subacute")
ACUTE_11            ("acute")
CHRONIC_11          ("chronic")
EPISODIC_11         ("periodic "|"episodic"|"binge"|"occasional")

FIRST_11            ((1st|first)([ ]|-))
SECOND_11           ((2nd|second)([ ]|-))
THIRD_11            ((3rd|third)([ ]|-))
FOURTH_11           ((4th|fouth)([ ]|-))
FIFTH_11            ((5th|fifth)([ ]|-))
SIXTH_11            ((6th|sixth)([ ]|-))
SEVENTH_11          ((7th|seventh)([ ]|-))
EIGHTH_11           ((8th|eighth)([ ]|-))
NINETH_11           ((9th|ninth|nineth)([ ]|-))
TENTH_11            ((10th|tenth)([ ]|-))
ELEVENTH_11         ((11th|eleventh)([ ]|-))
TWELVETH_11         ((12th|twelvth|twelveth)([ ]|-))
THIRTEENTH_11             ((13th|thirteenth)([ ]|-))
FOURTEENTH_11             ((14th|fourteenth)([ ]|-))
FIFTEENTH_11              ((15th|fifteenth)([ ]|-))
SIXTEENTH_11              ((16th|sixteenth)([ ]|-))

HEALING_00          ("healing")
HEALED_00           ("healed")
IMPROVING_00              ("improving")
IMPROVED_01         ("improved")
RESOLVED_00         ("resolved"|"resolve")
RESOLVING_00              ("resolving")

INDUCED_11          ("induced")
SIMPLE_REPAIR_01    ("simple repair")
REPAIRED_01         ("repaired")
REPAIRING_01              ("repairing")
REPAIR_01           ("repair")
REDUCED_01          ("reduced")
REDUCING_01         ("reducing")
REMOVED_10          ("removed")
UNTREATED_01              ("untreated")
TREATED_WITH_0011   ("treated with")
TREATED_01          ("treated")
UNSUCCESSFUL_10     ("unsuccessful")

ETIOLOGY            ("etiology"|"origin"|"nature")
ETIOLOGY_UNCERTAIN_01     ({ETIOLOGY}[ ]unclear|{ETIOLOGY}[
]uncertain|{ETIOLOGY}\\
[ ]considered|{ETIOLOGY}[ ]unspecified|question(able)?[
]{ETIOLOGY}|unsure\\
[ ]{ETIOLOGY}|unknown[ ]{ETIOLOGY}|unsure[ ]of[
]{ETIOLOGY}|unspecified\\
```

```
[ ]{ETIOLOGY}|uncertain[ ]{ETIOLOGY}|of[ ]uncertain[ ]{ETIOLOGY})
ETIOLOGY_10         ("etiology"|"in etiology")
ORIGIN_00           ("in origin")
ORIGIN_01           ("origin"|"due to")
NATURE_00           ("in nature")

DRUG_SCREEN_11            ("drug screen")
ACCU_CHECK_11             ("accu-check")
VDRL_11                   ("vdrl")

RIGHT_1                   ("right")
LEFT_1                    ("left")
BOTH_1                    ("both")
BILATERAL_1         ("bilateral")

AVOID__11           ("avoid")
CONSULTATION__01    ("consultation")
CONSULT__11         ("consult"|"consulted")
FOLLOWING_UP__1     (following([ ]|-)up)
FOLLOW_UP__01             ((follow-up|follow[
]up|followup|f\/u|F\/U|recheck|return)[ ]([^i][a-z\-]+[
]){0,5}(in|next)[ ])
RECHECK__11         (recheck[ ](if|after|when|next|in|with|on))
REFRAIN__11         ("refrain")
REFERRED__11              ("referred")
REFER__11           ("refer")
RETURNED__11              ("returned")
RETURN__11          ("return")
INSTRUCTED__11
("instructing"|"instructed"|"instructed"|"instruction"|"instructions"|"
says to"|"said to"|"told"|"suggest"|"suggestion"|"suggesting")
ADMONISHED__11            ("admonished")

WITH_0011           ("associated with"|"with associated"|"with ")
NOT_11_WITH_0011    ("without"|"not with "|"with none "|"with no ")

BUT_11                    ("but "|"although ")
HOWEVER_11          ("however ")

NO_01                     ("none ")
NOT_10                    ("not "|"never ")
NO_11                     ("no ")

PERIOD_00           (\.[ \t]*\n)
PERIOD_10           (\.[ \t]+)
PUNC_10                   (":"|";")
PUNC_11                   (",")

DASH                      ("-")
PERCENT                   ("%")
TERM                ([^ ,.:;/\n\t]+)

%%

{DISCARD}    { }

{DISCHARGE} { CPTCoder.add_parse_negation(); }
```

```
{ADMIT}          { CPTCoder.set_admit();
                   CPTCoder.add_parse_negation(); }
{SURGERY}        { CPTCoder.set_admit();
                   CPTCoder.add_parse_negation(); }
{FIELD_WORK}     { CPTCoder.add_parse_negation(); }
{RAD_READ1}      { CPTCoder.set_xray_disconfirm();
                   CPTCoder.add_parse_negation(); }
{RAD_READ2}      { CPTCoder.set_xray_disconfirm();
                   CPTCoder.add_parse_negation(); }
{ED_READ1}       { CPTCoder.set_xray_confirm(); }
{ED_READ2}       { CPTCoder.set_xray_confirm(); }
{DOCTOR}         { CPTCoder.add_parse_negation(); }
{PHY_ASST}       { CPTCoder.set_pa();
                   CPTCoder.add_parse_negation();
                 }
{SELF}           { CPTCoder.add_parse_text((char *)yytext, "myself");
                        /* need a function that will turn off a
negation set by {DOCTOR}
                           and will not allow a negation to be set by
{DOCTOR} for the
                           current phrase */ }
{PLAN}           { CPTCoder.add_parse_negation(); }
{CONSIDER}       { CPTCoder.add_parse_negation(); }
{CANCEL}         { CPTCoder.add_parse_negation(); }
{REFUSE}         { CPTCoder.add_parse_negation(); }
{WILL}           { CPTCoder.add_parse_negation(); }

{PATIENT}          { CPTCoder.add_parse_text((char *)yytext, "patient");
}

---T                    { yy_temporal = 1; }
T---                    { yy_temporal = 0; }
LT                      { if (yy_temporal)
CPTCoder.add_parse_temp_rel(-1); }
JUNK                    { if (yy_temporal) CPTCoder.add_parse_temp_rel(
0); }
MT                      { if (yy_temporal) CPTCoder.add_parse_temp_rel(
1); }
MN                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(1); }
HR                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(2); }
DA                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(3); }
WK                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(4); }
MO                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(5); }
YR                      { if (yy_temporal)
CPTCoder.add_parse_temp_scale(6); }

---S                    { yy_bodysurf = 1; }
S---                    { yy_bodysurf = 0; }
LT                      { if (yy_bodysurf)
CPTCoder.add_parse_bodysurf_rel(-1); }
JUNK                    { if (yy_bodysurf)
CPTCoder.add_parse_bodysurf_rel( 0); }
```

```
MT                      { if (yy_bodysurf)
CPTCoder.add_parse_bodysurf_rel( 1); }

---L                    { yy_measure = 1; }
L---                    { yy_measure = 0; }
MM                      { if (yy_measure)
CPTCoder.add_parse_size_scale(1); }
CM                      { if (yy_measure)
CPTCoder.add_parse_size_scale(2); }
IH                      { if (yy_measure)
CPTCoder.add_parse_size_scale(3); }

{POSSIBLE_11}           { CPTCoder.add_parse_probability(3); }
{POSSIBLE_10}           { CPTCoder.add_parse_probability(3); }
{POSSIBLE_01}           { CPTCoder.add_parse_probability(3); }
{PROBABLE_11}           { CPTCoder.add_parse_probability(4); }
{PROBABLE_10}           { CPTCoder.add_parse_probability(4); }
{RULE_OUT_11}           { CPTCoder.add_parse_probability(1); }
{RULE_OUT_10}           { CPTCoder.add_parse_probability(1); }

{RISK_OF_11}            { CPTCoder.add_parse_probability(2); }

{VS_0011}         { CPTCoder.add_parse_key(VS, "_0011_");
                      CPTCoder.add_parse_probability(1);  }

{NOT_IN_ER_01}          { CPTCoder.add_parse_evidence(2); }

{EVIDENCED_11}          { CPTCoder.add_parse_evidence(4); }
{EVIDENCE_01}           { CPTCoder.add_parse_evidence(4); }
{ALLEGED_00}            { CPTCoder.add_parse_evidence(3); }
{ALLEGED_11}            { CPTCoder.add_parse_evidence(3); }
{ATTEMPT}         { CPTCoder.add_parse_text((char *)yytext, "attempt");
                      CPTCoder.add_parse_negation();
                     /* dth - added 3/11/98 */ }
{DENIED_00}       { CPTCoder.add_parse_evidence(1);
                      CPTCoder.add_parse_negation(); }
{DENIED_11}       { CPTCoder.add_parse_evidence(1);
                      CPTCoder.add_parse_negation(); }

{STATUS_POST_11}  { CPTCoder.add_parse_text((char *)yytext, "status
post");
                      CPTCoder.add_parse_negation();
                     /* dth - negated on 3/11/98 */ }
{POST_11}         { CPTCoder.add_parse_text((char *)yytext, "post"); }
{HISTORY_OF_11}         { CPTCoder.add_parse_negation(); }
{HISTORY_OF_00}         { CPTCoder.add_parse_negation(); }
{OLD_11}          { CPTCoder.add_parse_key(NO, "_01_");
                      CPTCoder.add_parse_text((char *)yytext,
"no");
                      CPTCoder.add_parse_negation(); }

{SECONDARY_TO}          { CPTCoder.add_parse_diag_ordinality(2);
                      CPTCoder.add_parse_negation();
                      CPTCoder.add_parse_key(PERIOD, "_10_"); }
{SECONDARY_00}          { CPTCoder.add_parse_text((char *)yytext,
"secondary");
                      CPTCoder.add_parse_negation();
```

```
                        CPTCoder.add_parse_diag_ordinality(2);
                        CPTCoder.add_parse_key(PERIOD, "_10_"); }
{SECONDARY_11}          { CPTCoder.add_parse_text((char *)yytext,
"secondary");
                        CPTCoder.add_parse_negation();
                        CPTCoder.add_parse_diag_ordinality(2); }
{PRIMARY_11}            { CPTCoder.add_parse_text((char *)yytext,
"primary");
                        CPTCoder.add_parse_diag_ordinality(1);
                        CPTCoder.add_parse_severity(2); }
{ACUTE_11}        { CPTCoder.add_parse_text((char *)yytext, "acute");
                        CPTCoder.add_parse_severity(1); }
{CHRONIC_11}            { CPTCoder.add_parse_text((char *)yytext,
"chronic");
                        CPTCoder.add_parse_severity(3); }
{EPISODIC_11}           { CPTCoder.add_parse_text((char *)yytext,
"episodic");
                        CPTCoder.add_parse_severity(4); }

{FIRST_11}        { CPTCoder.add_parse_text((char *)yytext, "first");
                        CPTCoder.add_parse_ordinal(1); }
{SECOND_11}       { CPTCoder.add_parse_text((char *)yytext, "second");
                        CPTCoder.add_parse_ordinal(2); }
{THIRD_11}        { CPTCoder.add_parse_text((char *)yytext, "third");
                        CPTCoder.add_parse_ordinal(3); }
{FOURTH_11}       { CPTCoder.add_parse_text((char *)yytext, "fourth");
                        CPTCoder.add_parse_ordinal(4); }
{FIFTH_11}        { CPTCoder.add_parse_text((char *)yytext, "fifth");
                        CPTCoder.add_parse_ordinal(5); }
{SIXTH_11}        { CPTCoder.add_parse_text((char *)yytext, "sixth");
                        CPTCoder.add_parse_ordinal(6); }
{SEVENTH_11}            { CPTCoder.add_parse_text((char *)yytext,
"seventh");
                        CPTCoder.add_parse_ordinal(7); }
{EIGHTH_11}       { CPTCoder.add_parse_text((char *)yytext, "eighth");
                        CPTCoder.add_parse_ordinal(8); }
{NINETH_11}       { CPTCoder.add_parse_text((char *)yytext, "nineth");
                        CPTCoder.add_parse_ordinal(9); }
{TENTH_11}        { CPTCoder.add_parse_text((char *)yytext, "tenth");
                        CPTCoder.add_parse_ordinal(10); }
{ELEVENTH_11}           { CPTCoder.add_parse_text((char *)yytext,
"eleventh");
                        CPTCoder.add_parse_ordinal(11); }
{TWELVETH_11}           { CPTCoder.add_parse_text((char *)yytext,
"twelveth");
                        CPTCoder.add_parse_ordinal(12); }
{THIRTEENTH_11}         { CPTCoder.add_parse_text((char *)yytext,
"thirteenth");
                        CPTCoder.add_parse_ordinal(13); }
{FOURTEENTH_11}         { CPTCoder.add_parse_text((char *)yytext,
"fourteenth");
                        CPTCoder.add_parse_ordinal(14); }
{FIFTEENTH_11}          { CPTCoder.add_parse_text((char *)yytext,
"fifteenth");
                        CPTCoder.add_parse_ordinal(15); }
{SIXTEENTH_11}          { CPTCoder.add_parse_text((char *)yytext,
"sixteenth");
```

```
                        CPTCoder.add_parse_ordinal(16); }

{HEALING_00}            { CPTCoder.add_parse_status(101); }
{HEALED_00}       { CPTCoder.add_parse_status(111); }
{IMPROVING_00}          { CPTCoder.add_parse_status(102); }
{IMPROVED_01}           { CPTCoder.add_parse_status(112); }
{RESOLVING_00}          { CPTCoder.add_parse_status(103); }
{RESOLVED_00}           { CPTCoder.add_parse_status(113); }
{INDUCED_11}            { CPTCoder.add_parse_text((char *)yytext,
"induced");
                        CPTCoder.add_parse_status(6); }
{SIMPLE_REPAIR_01}      { CPTCoder.add_parse_text((char *)yytext,
"repair");
                        CPTCoder.add_parse_status(5); }
{REPAIRED_01}           { CPTCoder.add_parse_text((char *)yytext,
"repaired");
                        CPTCoder.add_parse_status(5); }
{REPAIRING_01}          { CPTCoder.add_parse_text((char *)yytext,
"repairing");
                        CPTCoder.add_parse_status(114); }
{REPAIR_01}       { CPTCoder.add_parse_text((char *)yytext, "repair");
                        CPTCoder.add_parse_status(5); }
{REDUCED_01}            { CPTCoder.add_parse_text((char *)yytext,
"reduced");
                        CPTCoder.add_parse_status(4); }
{REDUCING_01}           { CPTCoder.add_parse_text((char *)yytext,
"reducing");
                        CPTCoder.add_parse_status(115); }
{REMOVED_10}            { CPTCoder.add_parse_text((char *)yytext,
"removed");
                        CPTCoder.add_parse_status(3); }
{UNTREATED_01}          { CPTCoder.add_parse_status(1); }
{TREATED_WITH_0011}     { CPTCoder.add_parse_text((char *)yytext,
"treated with");
                        CPTCoder.add_parse_status(2); }
{TREATED_01}            { CPTCoder.add_parse_status(2); }
{UNSUCCESSFUL_10} { CPTCoder.add_parse_status(1); }

{ETIOLOGY_UNCERTAIN_01} { CPTCoder.add_parse_etiology(1); }
{ETIOLOGY_10}           { CPTCoder.add_parse_etiology(2); }
{ORIGIN_00}       { CPTCoder.add_parse_etiology(3); }
{ORIGIN_01}       { CPTCoder.add_parse_etiology(3); }
{NATURE_00}       { CPTCoder.add_parse_etiology(4); }

{DRUG_SCREEN_11}  { CPTCoder.add_parse_diagnostic(1);
                        CPTCoder.add_parse_text((char *)yytext,
"drug_screen"); }
{ACCU_CHECK_11}         { CPTCoder.add_parse_diagnostic(1);
                        CPTCoder.add_parse_text((char *)yytext,
"accu_check"); }
{VDRL_11}         { CPTCoder.add_parse_diagnostic(1);
                        CPTCoder.add_parse_text((char *)yytext,
"vrdl"); }

{RIGHT_1}         { CPTCoder.add_parse_loc(1); }
{LEFT_1}          { CPTCoder.add_parse_loc(0); }
{BOTH_1}          { CPTCoder.add_parse_text((char *)yytext, "both");
```

```
                        CPTCoder.add_parse_loc(2); }
{BILATERAL_1}           { CPTCoder.add_parse_text((char *)yytext,
"bilateral");
                        CPTCoder.add_parse_loc(2); }

{AVOID__11}       { CPTCoder.add_parse_text((char *)yytext, "avoid");
                        CPTCoder.add_parse_instruct(); }
{CONSULTATION__01}      { CPTCoder.add_parse_text((char *)yytext,
"consultation");
                        CPTCoder.add_parse_instruct(); }
{CONSULT__11}           { CPTCoder.add_parse_text((char *)yytext,
"consult");
                        CPTCoder.add_parse_instruct(); }
{FOLLOWING_UP__1} { CPTCoder.add_parse_text((char *)yytext, "following-
up");
                        CPTCoder.add_parse_instruct(); }
{FOLLOW_UP__01}         { CPTCoder.add_parse_text((char *)yytext,
"__follow-up");
                        CPTCoder.add_parse_negation();
                        CPTCoder.add_parse_instruct(); }
{RECHECK__11}           { CPTCoder.add_parse_text((char *)yytext,
"__recheck");
                        CPTCoder.add_parse_negation();
                        CPTCoder.add_parse_instruct(); }
{REFRAIN__11}           { CPTCoder.add_parse_text((char *)yytext,
"refrain");
                        CPTCoder.add_parse_instruct(); }
{REFERRED__11}          { CPTCoder.add_parse_text((char *)yytext,
"referred");
                        CPTCoder.add_parse_instruct(); }
{REFER__11}       { CPTCoder.add_parse_text((char *)yytext, "refer");
                        CPTCoder.add_parse_instruct(); }
{RETURNED__11}          { CPTCoder.add_parse_text((char *)yytext,
"returned");
                        CPTCoder.add_parse_instruct(); }
{RETURN__11}            { CPTCoder.add_parse_text((char *)yytext,
"return");
                        CPTCoder.add_parse_instruct(); }
{INSTRUCTED__11}  { CPTCoder.add_parse_text((char *)yytext,
"instruct");
                        CPTCoder.add_parse_instruct(); }
{ADMONISHED__11}  { CPTCoder.add_parse_text((char *)yytext,
"admonished");
                        CPTCoder.add_parse_instruct(); }

{NOT_11_WITH_0011}      { CPTCoder.add_parse_key(NO, "_11_");
                        CPTCoder.add_parse_text((char *)yytext, "not
with");
                        CPTCoder.add_parse_negation(); }
{BUT_11}          { CPTCoder.add_parse_key(BUT, "_11_"); }
{HOWEVER_11}            { CPTCoder.add_parse_key(HOWEVER, "_11_"); }

{NO_01}                 { CPTCoder.add_parse_text((char *)yytext,
"no");
                        CPTCoder.add_parse_key(NO, "_01_");
                        CPTCoder.add_parse_text((char *)yytext,
"no");
```

```
                        CPTCoder.add_parse_negation(); }
{NOT_10}          { CPTCoder.add_parse_text((char *)yytext, "not");
                        CPTCoder.add_parse_key(NO, "_10_");
                        CPTCoder.add_parse_text((char *)yytext,
"not");
                        CPTCoder.add_parse_negation(); }
{NO_11}                 { CPTCoder.add_parse_text((char *)yytext,
"no");
                        CPTCoder.add_parse_key(NO, "_11_");
                        CPTCoder.add_parse_text((char *)yytext,
"no");
                        CPTCoder.add_parse_negation(); }

{PERIOD_00}       { CPTCoder.add_parse_key(PERIOD, "_00_");
                        CPTCoder.add_break(); }
{PERIOD_10}       { CPTCoder.add_parse_key(PERIOD, "_10_");
                        CPTCoder.add_break(); }
{PUNC_10}         { CPTCoder.add_parse_key(PUNC, "_10_"); }
{PUNC_11}         { }

[\n]+                   { }
[ \t]+                  { }
{DASH}                  { }
{PERCENT}         { CPTCoder.add_parse_text((char *)yytext, "percent");
}
{NUMBER}          { if (yy_temporal) {

CPTCoder.add_parse_temp_cardinal(atof((char *)yytext));
                        }
                    else if (yy_bodysurf) {

CPTCoder.add_parse_bodysurf_cardinal(atof((char *)yytext));
                        }
                    else if (yy_measure) {
                            CPTCoder.add_parse_size(atof((char
*)yytext));
                        }
                        else {
                            CPTCoder.add_parse_cardinal(atof((char
*)yytext));
                            CPTCoder.add_parse_text((char *)yytext,
(char *)yytext);
                        }
                    }
{TERM}                  { CPTCoder.add_parse_text((char *)yytext, (char
*)yytext); }
EOT                     {
#ifdef LEXER
                        CPTCoder.parse_procedure();
#endif
                    }

%%

int yyposition() { return (strlen((char *)YY_CURRENT_BUFFER-
>yy_buf_pos)); }
```

```
#ifdef LEXER
CPTCoderClass CPTCoder;

int
main(int argc, char* argv[]) {
   return yylex();
}
#endif
```